My take on Variational Auto-Encoder

Tri Nguyen

nguyetr9@oregonstate.edu

August 17, 2023

Notation. $P(X = x)$ sometimes is written as $P(x)$ if the random variable X is clear from the context. Capital letter X denotes a random variable while small letter x denotes some realization of random variable.

1 Preliminary

Given two random variable X, Z, for arbitrary PDF $Q(Z)$, we have

$$\begin{aligned}
\log P(x) &= \log P(X = x) \\
&= \sum_z Q(z) \log P(x) \\
&= \sum_z Q(z) \log \left(\frac{P(z \mid x) P(x)}{P(z \mid x)} \right) \\
&= \sum_z Q(z) \log \left(\frac{Q(z)}{P(z \mid x)} \frac{P(x, z)}{Q(z)} \right) \\
&= \sum_z Q(z) \log \frac{Q(z)}{P(z \mid x)} + \sum_z Q(z) \log \frac{P(x, z)}{Q(z)} \\
&= D_{\mathsf{kl}}(Q(Z) \parallel P(Z \mid X = x)) + \mathcal{L}(Q), \qquad (1)
\end{aligned}$$

where we have defined

$$\mathcal{L}(Q) \triangleq \mathop{\mathbb{E}}_{Z \sim Q(Z)} \left[\log \frac{P(x, z)}{Q(z)} \right].$$

The decomposition in (1) is the beginning of variational inference. Two comments about that decomposition:

- It holds for any PDF $Q(Z)$. We will see that to make it related, people will choose that distribution as condition distribution $Q(Z \mid X)$, as we shall see in Section 2.
- As the first term $D_{\mathsf{kl}}(\parallel)$ is nonnegative, we have a lower of the likelihood [1]

  $$\log P(x) \geq \mathcal{L}(Q).$$

  Admittedly, this is a very superficial way to explain the lower bound. We should have derived the lower bound from something like Jensen's inequality. However, I find this way easier and more intuitive to remember.
- The lower bound $\mathcal{L}(Q)$ is called the *variational lower bound*. It depends on two functions and one realization: PDF $Q(X)$, PDF $P(X, Z)$, and x, resp. It is written as $\mathcal{L}(Q)$ only emphasize that Q plays as variable although it is a function. Hence the term variational.

[1]The term likelihood used to refer $P(X)$ might not very sensible, as there is no parameter. However, in almost every context, that $P(X)$ will belong to some parametric family.

- And lastly, $\log P(x)$ reaches its lower bound only if $D_{\mathsf{kl}}(Q \mid P \parallel \ =)0$. This means $\log P(x)$ reaches its minimum if variable $Q(Z)$ equals to the posterior $P(Z \mid X)$. We wish to find an good approximation of the true posterior distribution. Yet, it is obvious that we could not just set $Q(Z)$ to the true posterior since then the optimization on $\mathcal{L}(Q)$ will be intractable. It is intractable because we are taking expectation over $Q(Z)$ which is $P(Z \mid)$ in this case.

2 Main Paper

2.1 Problem setting

$$\begin{tikzcd} \boldsymbol{Z} \arrow[rr, "\boldsymbol{X} \sim p_{\boldsymbol{\theta}^\star}(\boldsymbol{X} \mid \boldsymbol{Z})"] & & \boldsymbol{X} \end{tikzcd}$$

Figure 2.1: VAE generative model. The conditional distribution family $p_{\boldsymbol{\theta}}(\boldsymbol{X} \mid \boldsymbol{Z})$ is assumed to be known, such as Gaussian, while the true $\boldsymbol{\theta}^\star$ is unknown. $\boldsymbol{X}$ is observed variable while $\boldsymbol{Z}$ is hidden variable.

Block 1. Given:

- Function family of $p_{\boldsymbol{\theta}^\star}(\boldsymbol{Z})$
- Function family of $p_{\boldsymbol{\theta}^\star}(\boldsymbol{X} \mid \boldsymbol{Z})$
- And N i.i.d. samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, each $\boldsymbol{x}_i \sim p_{\boldsymbol{\theta}^\star(\boldsymbol{X} \mid \boldsymbol{Z})}$

Goal: Identify $\boldsymbol{\theta}^\star$.

Uniqueness of $\boldsymbol{\theta}^\star$?

Naive attempt. Let's use maximum likelihood (ML) principle to estimate $\boldsymbol{\theta}^\star$. Log likelihood for sample $\boldsymbol{x}^i$ is

$$\begin{aligned} \log \mathsf{Pr}(\boldsymbol{x}_i) &= \log \left(\sum_{\boldsymbol{z}} \mathsf{Pr}(\boldsymbol{x}_i, \boldsymbol{z}) \right) \\ &= \log \left(\sum_{\boldsymbol{z}} \mathsf{Pr}(\boldsymbol{x}_i \mid \boldsymbol{z}) \mathsf{Pr}(\boldsymbol{z}) \right) \\ &= \log \left(\sum_{\boldsymbol{z}} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i \mid \boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) \right) \end{aligned}$$

Since we know function family of $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{z})$, we can plug them in here, and then use Pytorch to optimize with respect to $\boldsymbol{\theta}$. Unfortunately, life is not that easy. We have no way to express that summation over $\boldsymbol{z}$ in a close form expression. And it is impossible to directly enumerate all possible $\boldsymbol{z}$ to construct the objective function. And it gets worse if $\boldsymbol{z}$ is continuous, ...

Prior attempt. Okay, so there is no way to deal with MLE objective directly. Then lets try to optimize its surrogate instead. People have used the lower bound of negative MLE as surrogate for ages. As covered in Section 1, for arbitrary proper distribution $q(\boldsymbol{Z})$,

$$\log \mathsf{Pr}(\boldsymbol{x}_i) = D_{\mathsf{kl}}(q(\boldsymbol{Z}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)) + \mathop{\mathbb{E}}_{q(\boldsymbol{Z})} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{Z}, \boldsymbol{x}_i)}{q(\boldsymbol{Z})} \right] \geq \mathop{\mathbb{E}}_{q(\boldsymbol{Z})} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{Z}, \boldsymbol{x}_i)}{q(\boldsymbol{Z})} \right] \tag{2}$$

There are 2 things we can do with this inequality: (i) look for a $q(\boldsymbol{Z})$ to obtain a lower bound as tight as possible (ii) and then maximize it. For task (i), we should $q(\boldsymbol{Z})$ as $q(\boldsymbol{Z} \mid \boldsymbol{x}_i)$ as the true posterior depending on $\boldsymbol{x}_i$. The functional family of $q(\boldsymbol{Z} \mid \boldsymbol{x}_i)$ should be as much flexible as possible so that lower bound is close to the true objective. But we also needs to restrict it since choosing a too complicated distribution will lead to intractability. To see this, assuming choosing $q(\boldsymbol{Z} \mid \boldsymbol{x}_i) = p_{\boldsymbol{\theta}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)$, i.e., the true posterior[1]. The lower bound is now equal to the true likelihood and is

$$\mathcal{L}(Q) = \mathcal{L}(p_{\boldsymbol{\theta}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)) = \mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}}(\boldsymbol{Z}|\boldsymbol{x}_i)} [\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)] = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i),$$

This is meaningless as it returns to the original MLE. And as shown in the native attempt, this wouldn't work.

Block 2. This shows the tension in choosing the functional family to approximate true posterior while keeping it tractable. And we shall see how people deal with this situation.

Okay, lets restrict $Q(\boldsymbol{Z} \mid \boldsymbol{x}_i)$ to something more tractable. What about

$$Q(\boldsymbol{Z} \mid \boldsymbol{x}_i) = \prod_{j=1}^{k} Q_j(\boldsymbol{Z}_j \mid \boldsymbol{x}_i), \tag{3}$$

where we partition elements in $\boldsymbol{Z}$ to $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_k$. This is developed in physics and called *mean field theory*. Plug in to and do some manipulation would lead to a system of equations. And then based on that, they derive a iterative method ...

fill this in later, or not.

At the end of the day, the assumption in (3) might not true in lots of cases. If that happens, even if we get to the optimal solution, it is still not the optimum in terms of MLE objective.

2.2 The VAE way

As everything is approximated by neural network, why don't we use neural network to construct the approximate posterior family. In particular, the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)$ might be very complicated, so let the deep neural network deal with it. We choose $q(\boldsymbol{Z} \mid \boldsymbol{x}_i)$ in (2) as $q_{\boldsymbol{\phi}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)$, where $q_{\boldsymbol{\phi}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)$ must satisfy:

- Be a proper distribution. This is actually not easy, and was not emphasized enough in the paper. It is deal by choosing some known distribution, such as Gaussian, exponential distribution, and let certain parameters being controlled by the neural network.
- The key parameter is then determined by $f_{\boldsymbol{\phi}}(\boldsymbol{x}_i)$, where $f_{\boldsymbol{\phi}}(\boldsymbol{x}_i)$ is a neural network.

The hope is that the posterior might belong the some well known distribution. This could be true if the prior and the condition distribution are chosen in a certain ways (conjugate distributions for instance). **But it would not hold in general, especially in case we do not even know the prior**. To be fair, assume the distribution of the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)$ belongs to some well-known distribution, estimating the parameters of this distribution is still very hard. We only know that these parameters might depend on $\boldsymbol{x}_i$. That is the focus of VAE. It proposal to use a neural network to approximate this complicated unknown dependence of $\boldsymbol{Z}$ on $\boldsymbol{x}_i$.

Toward this end, we have

$$\log p_{\boldsymbol{\theta}}(x^{(i)}) = D_{\mathsf{kl}}(q_{\boldsymbol{\phi}}(Z \mid x^{(i)}) \parallel p_{\boldsymbol{\theta}}(Z \mid x^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; x^{(i)}), \quad \text{(Equation (1) in [2])}$$

[1]Note that we **never** have closed-form expression for this $p_{\boldsymbol{\theta}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)$, but that's okay we might not need to know it.

where

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; x^{(i)}) &\triangleq \mathop{\mathbb{E}}_{Z \sim Q(Z|x^{(i)})} \left[\log \frac{P(x^{(i)}, Z)}{Q(Z \mid x^{(i)})} \right] \\
&= \mathop{\mathbb{E}}_{Z \sim Q(Z|x^{(i)})} \left[\log \frac{p_{\boldsymbol{\theta}}(x^{(i)}, Z)}{q_{\boldsymbol{\phi}}(Z \mid x^{i})} \right] \\
&= \mathop{\mathbb{E}}_{Z \sim q_{\boldsymbol{\phi}}(Z|x^{(i)})} \left[-\log q_{\boldsymbol{\phi}}(Z \mid x^{(i)}) + \log p_{\boldsymbol{\theta}}(x^{(i)}, Z) \right] \quad \text{(RHS of (2) in [2])} \\
&= -D_{\mathsf{kl}}(q_{\boldsymbol{\phi}}(\boldsymbol{Z} \mid \boldsymbol{x}_i) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathop{\mathbb{E}}_{\boldsymbol{Z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{x}_i)} [\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i \mid \boldsymbol{Z})]
\end{aligned}$$

We again want to maximize the lower bound, but now it is with respect to both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. By changing $\boldsymbol{\phi}$, it is allowed to find a tighter lower bound, while by changing $\boldsymbol{\theta}$, it is allowed to maximize this lower bound.

The notation $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; x^{()})$ reflects our last comment:

- The variational lower bound depends on $P(X, Z)$ via $\boldsymbol{\theta}$. There is a subtle point thought. The dependence of $\mathcal{L}$ on $\boldsymbol{\theta}$ is only "complete" if the prior $P(Z)$ is fixed. In their Section 3, they do use a fixed prior for Z.
- The variational lower depends on $Q(Z)$ via $\boldsymbol{\phi}$.
- Lastly, it also depends on particular realization $x^{(i)}$.

Then for the whole dataset,

$$\sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) = \sum_{i=1}^{n} D_{\mathsf{kl}}(q_{\boldsymbol{\phi}}(Z \mid \boldsymbol{x}^{(i)}) \parallel p_{\boldsymbol{\theta}}(Z \mid \boldsymbol{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}),$$

where

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}) = \sum_{i=1}^{n} \left(-D_{\mathsf{kl}}(q_{\boldsymbol{\phi}}(\boldsymbol{Z} \mid \boldsymbol{x}_i) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathop{\mathbb{E}}_{\boldsymbol{Z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{x}_i)} [\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i \mid \boldsymbol{Z})] \right)$$

We want to optimize this objective with respect to both $\boldsymbol{\theta}, \boldsymbol{\phi}$ simultaneously using gradient descent (Pytorch). The gradient of the first term can be evaluated "easily". If we carefully choose prior and the approximate posterior, the KL divergence can be expressed in close-form. The trickery occurs when evaluating gradient of the second term with respect to $\boldsymbol{\phi}$ as it appears under the expectation. In general, we are dealing with something like

$$\nabla_{\boldsymbol{\phi}} \mathop{\mathbb{E}}_{Z \sim f_{\boldsymbol{\phi}}(Z)} [g(Z)]$$

The native attempt was presented in [3]. Gradient w.r.t $\boldsymbol{\phi}$ can be estimated as

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathop{\mathbb{E}}_{Z \sim f_{\boldsymbol{\phi}}(Z)} [g(Z)] &= \nabla_{\boldsymbol{\phi}} \int_Z f_{\boldsymbol{\phi}}(Z) g(Z) dZ \\
&= \int_Z g(Z) \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(Z) dZ \quad \text{(Leibniz integral rule)} \\
&= \int_Z g(Z) f_{\boldsymbol{\phi}}(Z) \nabla_{\boldsymbol{\phi}} \log f_{\boldsymbol{\phi}}(Z) dZ \\
&= \mathop{\mathbb{E}}_{Z \sim f_{\boldsymbol{\phi}}(Z)} [g(Z) \nabla_{\boldsymbol{\phi}} \log f_{\boldsymbol{\phi}}(Z)] \\
&\approx \frac{1}{L} \sum_{i=1}^{L} g(z_i) \nabla_{\boldsymbol{\phi}} \log f_{\boldsymbol{\phi}}(z_i)
\end{aligned}$$

Is this the same with policy gradient?

However, this gradient estimator exhibits high variance. To get a better estimator, VAE introduced the well-known reparameterization trick. Note that $\boldsymbol{Z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)$ is chosen as some well-known distribution family, such as Gaussian, it is then possible to find $g_{\boldsymbol{\phi}}(\cdot)$ to reparameterize distribution of $\boldsymbol{Z}$ as

$$\widetilde{\boldsymbol{Z}} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}_i), \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}).$$

Essentially, we decompose $\boldsymbol{Z}$ into 2 parts: a deterministic mapping $g_{\boldsymbol{\theta}}(\cdot)$ that maps $\boldsymbol{x}_i$ to a certain $\boldsymbol{Z}$. And a stochastic part which is accounted by $p(\boldsymbol{\epsilon})$. Of course, this decomposition is not always feasible. But when it is possible, we get a better gradient estimator since

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_i) &= \mathop{\mathbb{E}}_{\boldsymbol{Z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{Z} \mid \boldsymbol{x}_i)} \left[-\log q_{\boldsymbol{\phi}}(\boldsymbol{Z} \mid \boldsymbol{x}_i) + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{Z}) \right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})} \left[-\log q_{\boldsymbol{\phi}}(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}_i) \mid \boldsymbol{x}_i) + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}_i)) \right]\end{aligned}$$

Then the gradient with respect to $\boldsymbol{\phi}$ is estimated as

$$\begin{aligned}\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_i) &= \mathop{\mathbb{E}}_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})} \left[-\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}_i) \mid \boldsymbol{x}_i) + \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}_i)) \right] \\ &\approx \frac{1}{L} \sum_{\ell=1}^{L} \left[-\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}_\ell, \boldsymbol{x}_i) \mid \boldsymbol{x}_i) + \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}_\ell, \boldsymbol{x}_i)) \right],\end{aligned}$$

where we sample ℓ such $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. And Pytorch can work on this.

3 Diffusion things

Let $\boldsymbol{x}$ follow the data distribution $p_0(\boldsymbol{x})$. We diffuse $\boldsymbol{x}$ by repetitively adding a little Gaussian noise, i.e.,

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \sigma_t^2), \quad t = 1, .., T$$

In the limit of $T \to \infty$ and $\sigma_t \to 0$, this process will lead to $\boldsymbol{x}_T \sim \mathcal{N}(0,)$, a completely white noise.

Given $\boldsymbol{x}_0$ follow some distribution p_0, i.e., $\boldsymbol{x}_0 \sim p_0(\boldsymbol{x})$, and a list of transformations, each of which take an $\boldsymbol{x} \in \mathcal{X}$ and produce a $\boldsymbol{x}' \in \mathcal{X}$. Applying these transformations consecutively to obtain:

$$\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \ldots \to \boldsymbol{x}_T$$

This process takes input as $\boldsymbol{x}_0$ and produces a list of random variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$. In a more abstract sense, we can think of this process $\mathcal{F}$ is defined by the list of transformations, taking the input as a distribution p_0, and producing outputs as a list of distributions $p_0, p_1, \ldots, p_T$ (including p_0 for complete), which is called a probability path. The only thing that I assume is: given p_0, a fixed F will always produce a same probability path $p_0, p_1, \ldots, p_T$.

Question is: Does there exist an reverse process $\mathcal{F}'$ such that

$$\mathcal{F}(p_0) = \mathcal{F}'(p_T).$$

Probably. The real question is how hard to construct such $\mathcal{F}'$.

Now, let's make thing concrete. If $\mathcal{F}$ is a diffusion process then the reverse process $\mathcal{F}'$ is also a diffusion process [1].

References

[1] Brian DO Anderson. "Reverse-time diffusion equation models". In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326.

[2] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[3] John Paisley, David Blei, and Michael Jordan. "Variational Bayesian inference with stochastic search". In: *arXiv preprint arXiv:1206.6430* (2012).